Move Semantics Solutions

Copyright © James Raynard 2021

Value Semantics

- Describe some situations where the compiler adds copy operations in traditional C++
  - Passing argument by value
  - Returning by value
  - Populating standard containers
- Are there any ways these can safely be avoided?
  - Instead of passing by value, we can pass by const reference

Copyright © James Raynard 2021

Value Semantics

- Why are these copy operations considered undesirable?
  - Copying can create a lot of overhead
  - e.g. vector of strings - allocate new memory buffer for vector, allocate new memory buffer for each string, copy data into each string, copy each string into vector
  - If the copied object will not be used again, this overhead is wasteful and inefficient
- Why do other languages do not have this problem?
  - Most languages use references which are managed by a garbage collector
  - In C, anything more complex than copying members requires an explicit function call

Copyright © James Raynard 2021

Value Semantics

- What techniques does traditional C++ provide for reducing the copying overhead?
  - Copy elision when returning from function calls
  - Overloading swap()

Copyright © James Raynard 2021

Move Semantics

- Explain briefly what is meant by "move semantics" in Modern C++
  - Move semantics is an optimization for copying objects
  - If the source object is an rvalue, its data can be moved into the target instead of being copied
- Describe what happens what the following operations are performed on two variables:
  - Copying - A copy of the source's data is made and used to populate the target
  - Swapping - the source and the target exchange their data without copying
  - Moving - the source's data is transferred to the target without being copied

Copyright © James Raynard 2021

rvalue

- Under what circumstances is it safe to move data from a object?
  - When the moved-from object will not be used again
  - Literal value
  - Temporary object
  - Object that we have indicated we do not want to use again
- What term is used to describe an object in these circumstances?
  - The object is said to be an rvalue

Copyright © James Raynard 2021

Pass by move

- If we have a function which takes an argument by value, in what circumstances will an object be moved into this argument?
  - The object is an rvalue
  - The object's type is moveable
- What happens if these circumstances do not apply?
  - The object will be copied into the argument as usual
- Is pass by move compatible with existing code written for older versions of C++?
  - Yes. If an rvalue is not moveable, it will be copied as before
  - If an rvalue is moveable, it will be moved without needing to make any change to the existing code

Copyright © James Raynard 2021

Moving a function call argument

- A function can specify that an argument can only be moved into
- Why is this useful?
  - We can write overloaded functions that behave differently, depending on whether the argument was moved or copied
- What happens if the function is passed an object that cannot be moved?
  - The function cannot be called with that argument
  - The compiler will look for an overload that does not require the variable to be moved
  - If it cannot find one, it will issue an error

Copyright © James Raynard 2021